WEST

Freeform Search

Database: US Patents Full-Text Database
US Pre-Grant Publication Full-Text Database
JPO Abstracts Database
EPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term:

Display: 50 **Documents in Display Format:** KWIC **Starting with Number** 1

Generate: ○ **Hit List** ◉ **Hit Count** ○ **Side by Side** ○ **Image**

Search | Clear | Help | Logout | Interrupt

Main Menu | Show S Numbers | Edit S Numbers | Preferences | Cases

Search History

DATE: Monday, October 21, 2002 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=ADJ			
L10	l7 and l2	5	L10
L9	L8 and l3	7	L9
L8	L7 and select$3 near2(means or unit or block or section or circuit$3)	7	L8
L7	L6 same adder	27	L7
L6	L3 same multiplier	83	L6
L5	L3 same multiplier same select$3 near2(means or unit or block or section or circuit$3)	1	L5
L4	L3 same multiplier same select$3 adj2(means or unit or block or section or circuit$3)	1	L4
L3	(plurality or multiple or more than one or various) near3 operand	1368	L3
L2	accumulat$3 near2 (buffer or register or memory)	11481	L2
L1	accumulat$3 near2 (buffer or register)	8311	L1

WEST

Generate Collection | Print

Search Results - Record(s) 1 through 7 of 7 returned.

☐ 1. Document ID: US 6408376 B1

L9: Entry 1 of 7 File: USPT Jun 18, 2002

DOCUMENT-IDENTIFIER: US 6408376 B1
TITLE: Method and apparatus for instruction set architecture to perform primary and shadow digital signal processing sub-instructions simultaneously

Abstract Text (1):
Disclosed is a method, apparatus, and an instruction set architecture (ISA) for an application specific signal processor (ASSP) tailored to digital signal processing (DSP) applications. The instruction set architecture implemented with the ASSP, is adapted to DSP algorithmic structures. In one embodiment, a single DSP instruction includes a pair of sub-instructions: a primary DSP sub-instruction and a shadow DSP sub-instruction. Both the primary and the shadow DSP sub-instructions are dyadic DSP instructions performing two operations in one instruction cycle. The DSP operations, in one embodiment, include a multiply instruction (MULT), an addition instruction (ADD), a minimize/maximize instruction (MIN/MAX), and a no operation instruction (NOP). Each signal processing unit includes a primary stage to execute a primary DSP sub-instruction based upon current data and a shadow stage to simultaneously execute a shadow DSP sub-instruction based upon delayed data stored locally within registers of the signal processing units. Control logic is utilized to control shadow selectors of each signal processing unit to select delayed data (specified by the shadow DSP sub-instruction) for use by the shadows stages of the signal processing units. In this way, the present invention efficiently executes DSP instructions by simultaneously executing primary DSP sub-instructions (based upon current data) and shadow DSP sub-instructions (based upon delayed locally stored data) with a single DSP instruction thereby performing four operations per single instruction cycle.

Drawing Description Text (24):
FIG. 9a illustrates the architecture of a data typer and aligner (DTAB) of a signal processing unit (SP2) to select current data for a primary stage and delayed data for use by a shadow stage from the x bus according to one embodiment of the present invention.

Drawing Description Text (25):
FIG. 9b illustrates the architecture of a data typer and aligner (DTAB) of a signal processing unit (SP2) to select current data for a primary stage and delayed data for use by a shadow stage from the y bus according to one embodiment of the present invention.

Detailed Description Text (105):
FIG. 9a illustrates the architecture of a data typer and aligner (DTAB) 502 of a signal processing unit 300 to select current data for the primary stage 561 and delayed data for use by the shadow stage 562 of an SP from the x bus 531. Particularly, FIG. 9a illustrates DTAB 502C of SP2300C (shown in FIG. 7) to select source value SX.sub.2 for output to the primary stage 561, as specified by the primary DSP sub-instruction, and to select shadow value SHX.sub.2 from delayed data, x' and x", for output to the shadow stage 562 as specified by the shadow DSP sub-instruction.

Detailed Description Text (111):
FIG. 9b illustrates the architecture of a data typer and aligner (DTAB) 502 of a signal processing unit 300 to select current data for the primary stage 561 and delayed data for use by the shadow stage 562 of an SP from the y bus 533. Particularly, FIG. 9b illustrates DTAB 502C of SP2300C (shown in FIG. 7) to select source value SY.sub.2 for output to the primary stage 561, as specified by the primary DSP sub-instruction, and

to select shadow value SHY.sub.2 from delayed data, y' and y", to output to the shadow stage 562 as specified by the shadow DSP sub-instruction.

CLAIMS:

9. The signal processor of claim 5 for performing digital signal processing instructions, the signal processor further comprising:

a reduced instruction set computer (RISC) control unit and a pipeline controller to predecode the primary and shadow digital signal processing sub-instructions into a plurality of preliminary instruction execution signals; and

wherein the at least one signal processing unit further includes

a plurality of final decoders coupled to a plurality of multiplexers, each of the first and second adders and first and second multipliers having an input multiplexer from the plurality of multiplexers to receive operands responsive to the selection by those of the plurality of final decoders coupled thereto.

☐ 2. Document ID: US 6397238 B2

L9: Entry 2 of 7 File: USPT May 28, 2002

DOCUMENT-IDENTIFIER: US 6397238 B2
TITLE: Method and apparatus for rounding in a multiplier

Brief Summary Text (15):
In one embodiment, the multiplier may comprise a partial product generator, a selection logic unit, and an adder. The multiplier may also comprise a multiplicand input configured to receive a multiplicand operand (signed or unsigned), a multiplier input configured to receive a multiplier operand (also signed or unsigned), and a sign-in input. The sign-in input is configured to receive a sign-in signal indicative of whether the multiplier is to perform signed or unsigned multiplication. The partial product generator, which is coupled to the multiplicand input, is configured to generate a plurality of partial products based upon the multiplicand operand. The selection logic unit, which is coupled to the partial product generator and the multiplier input, is configured to select a number of partial products from the partial product generator based upon the multiplier operand. The adder, which is coupled to the selection logic unit, is configured to sum the selected partial products to form a final product. The final product, which may be signed or unsigned, may then be output to other parts of the microprocessor.

Brief Summary Text (18):
In another embodiment, the multiplier may be capable of multiplying one pair of N-bit operands or two pairs of N/2-bit operands simultaneously. The multiplier may comprise a multiplier input and a multiplicand input, each configured to receive an operand comprising one N-bit value or two N/2-bit values. The multiplier may also comprise a partial product generator coupled to the multiplicand input, wherein the partial product generator is configured to generate a plurality of partial products based upon the value of the multiplicand operand. The multiplier may further comprise a selection logic unit coupled to the partial product generator and the multiplier input. The selection logic unit may be configured to select a plurality of partial products from the partial product generator based upon the value of the multiplier operand. An adder may be coupled to the selection logic unit to receive and sum the selected partial products to form a final product comprising either one 2N-bit value or two N-bit values. The multiplier may receive a vector_in signal indicating whether vector or scalar multiplication is to be formed.

Brief Summary Text (24):

A method for rounding and normalizing results within a multiplier is also contemplated. In one embodiment, the method comprises multiplying a first operand and a second operand to form a plurality of redundant-form components. A rounding constant is generated and added to the redundant-form component in two different bit positions. The first position assumes an overflow will occur, while the second position assumes no overflow will occur. A particular set of bits are selected for output as the final result from either the first addition or the second addition.

Detailed Description Text (29):
Turning now to FIG. 6, more detail of one embodiment of multiplier 50 is shown. In this embodiment, multiplier 50 comprises a partial product generator 60, a partial product selection logic unit 62, and an adder 64. As shown in the figure, partial product generator 60 is coupled to selection logic unit 62, which is in turn coupled to adder 64. When one of execution units 36C-36E receives an instruction invoking the multiplication function, the execution unit conveys two operands to multiplier 50, i.e., a multiplicand operand 72 and a multiplier operand 74. Partial product generator 60 is coupled to receive multiplicand operand 72, which is used as a starting value for calculating a plurality of partial products 70. For example, if partial product generator 60 is configured to use the 2-bit version of Booth's algorithm, the following partial products would be generated: the multiplicand itself ("+M"), a shifted version of the multiplicand ("+2M"), an inverted version of the multiplicand ("-M"), a shifted and inverted version of the multiplicand ("-2M"), and two constants, i.e., a positive zero ("+0") and a negative zero ("-0") in two's complement form.

Detailed Description Text (30):
Partial product selection unit 62 is coupled to receive multiplier operand 74. Selection unit 62 is configured to select a number of partial products from generator 60 based upon particular fields within multiplier operand 74. For example, using the 2-bit version of Booth's algorithm, multiplier operand 74 is padded with leading and trailing zeros (assuming an unsigned multiplication is being performed), and then one partial product is selected by each 3-bit field within the operand.

Detailed Description Text (31):
Finally, adder 64 is configured to receive and sum the partial products selected by selection unit 62. As noted in the figure, the selected partial products 68 are shifted before they are summed. The resulting final product 76 is output to the execution unit that transmitted the operands. As previously noted, multiplier 50 may advantageously be configured to perform both signed and unsigned multiplication. This is described in greater detail below.

Detailed Description Text (33):
Turning now to FIG. 7, details of one embodiment of multiplier 50 are shown. The figure also illustrates the operation of multiplier 50 for an unsigned multiplication. While the figure shows an 8-bit by 8-bit multiplier using the 2-bit version of Booth's algorithm, other configurations are possible and contemplated, e.g., a 32-bit by 32-bit multiplier using a 3-bit version of Booth's algorithm. In this embodiment, multiplier 50 further comprises a "sign-in" input 78, which indicates whether a signed or unsigned multiplication is to be performed. Sign-in input 78 is coupled to AND gate 86A, which also receives the most significant bit ("MSB") of multiplier operand 74. AND gate 86A outputs an "effective sign" bit 90 for multiplier operand 74 which is copied and appended to multiplier operand 74 for use by selection logic unit 62. Sign-in input 78 is also routed to AND gate 88B, which similarly calculates and appends an effective sign bit 92 for multiplicand operand 72. While other effective sign calculation logic may be used, the configuration illustrated advantageously generates an effective sign of zero for all unsigned operands and positive signed operands using a minimum amount of logic. Furthermore, in the embodiment shown only signed negative operands receive an asserted effective sign bit.

Detailed Description Text (50):
In this embodiment, adder 64 comprises three pluralities of multiplexers 160A-160D, 162A-162E, and 164C-164E. Multiplexers 160A-160D are controlled by vector_in signal 120 and operate to "zero-out" portions of the partial products to prevent corruption of the vector components within final product 76 during the summation within adder 64. Multiplexers 164C-E are also controlled by vector_in signal 120 and operate to select either extra constant bits 140C-140E (in the event of a vector multiplication) or a zero constant (in the event of a scalar multiplication) for addition into the more significant product. Multiplexers 162A-162D are controlled by sign_in input 78 and are configured to select either the effective sign bit of the more significant portion of the selected partial product (in the event of a signed vector multiplication) or the

actual sign (in the event of an unsigned vector multiplication). Multiplexers 164C-164E are also controlled by vector_in signal 102 and perform the same function as multiplexers 310B, 312B, 314B, and 316B, i.e., they select a constant zero instead of extra constant bit S2 if scalar multiplication is performed. Note that other configurations of logic for zeroing out and partial product selection are possible and contemplated. Further note that multiplexers 160A-160D, 162A-162E, and 164C-164E may be configured as part of adder 64, selection logic unit 62, or as a separate part of multiplier 50.

Detailed Description Text (79):
Next, the desired number of upper bits from the outputs of LSB fix-up logic units 288A and 288B may be conveyed to multiplexer 290, which selects one of the two values (overflow or no overflow) as output 292. Multiplexer 290 may be controlled by MSB 284 from the output of fix-up logic unit 288A. By looking at the most significant bit, a determination of whether an overflow occurred can be made. If an overflow occurred, the upper bits from the output of LSB fix-up logic unit 288A are selected. If an overflow did not occur, the upper bits from the output of LSB fix-up logic unit 288B are selected. Note that other control configurations are also possible, e.g., MSB 284 may be the most significant bit of the output from fix-up logic unit 288B. Furthermore, in some embodiments of multiplier 50 only one fix-up logic unit may be needed. For example, the single fix-up logic unit may be coupled to the output of multiplexer 290 and perform the fix-up before final result 292 is output.

CLAIMS:

1. A multiplier capable of supporting different rounding modes comprising:

a multiplier input configured to receive a multiplier operand;

a multiplicand input configured to receive a multiplicand operand;

a partial product generator configured to receive said multiplicand operand and generate a corresponding plurality of potential partial products;

partial product selection logic configured to receive said multiplier operand and select at least one of said potential partial products;

a partial product array adder coupled to partial product selection logic and configured to receive and sum said selected partial products to form a redundant-form product;

rounding constant selection logic configured to select a rounding constant corresponding to the selected rounding mode;

a first path comprising one or more adders, wherein said first path is configured to receive, sum, and round said redundant-form product and said rounding constant, assuming no overflow will occur, to form a first non-redundant-form product;

a second path comprising one or more adders, wherein said second path is configured to receive, sum, and round said redundant-form product and said rounding constant, assuming an overflow will occur, to form a second non-redundant-form product; and

selection logic coupled to said first path and said second path, wherein said selection logic is configured to select between said first non-redundant-form product and said second non-redundant-form product.

10. The multiplier as recited in claim 9, wherein said multiplier further comprises a plurality of adders coupled to said partial product array and configured to sum said vector component results to form the vector dot product of said multiplier and multiplicand operands, wherein said multiplier and multiplicand operands each comprise a plurality of vector components.

16. A microprocessor comprising:

an instruction cache configures to receive and store instructions,

a multiplier capable of supporting different rounding modes, wherein the multiplier is configured to receive and execute multiply instructions from the instruction cache, wherein the multiplier comprises:

a means for generating partial products, wherein the means for generating partial products is configured to receive a multiplicand operand and generate a corresponding plurality of potential partial products;

a means for selecting partial products, wherein the means for selecting partial products is configured to receive a multiplier operand and select at least one of the potential partial products based on the multiplier operand;

a means for adding, wherein the means for adding is coupled to the means for selecting and is configured to receive and sum the selected partial products to form a redundant-form product;

a means for selecting rounding constants, wherein the means for selecting rounding constants is configured to select a particular rounding constant that corresponds to a selected rounding mode;

a first path comprising one or more adders, wherein the first path is configured to receive, sum, and round the redundant-form product and the rounding constant, assuming no overflow will occur, to form a first non-redundant-form product;

a second path comprising one or more adders, wherein the second path is configured to receive, sum, and round the redundant-form product and the rounding constant, assuming an overflow will occur, to form a second non-redundant-form product; and

selection logic coupled to the first path and the second path, wherein the selection logic is configured to select between the first non-redundant-form product and the second non-redundant-form product.

☐ 3. Document ID: US 6393452 B1

L9: Entry 3 of 7 File: USPT May 21, 2002

DOCUMENT-IDENTIFIER: US 6393452 B1
TITLE: Method and apparatus for performing load bypasses in a floating-point unit

Brief Summary Text (10):
The apparatus of the present invention comprises a floating-point unit which comprises a register file, at least one bypass component and at least one multiply accumulate unit. The register file comprises a plurality of registers for storing operand data to be operated on and for storing results of operations that have been performed by the floating-point unit. The bypass component is configured to perform a memory format-to-register format conversion. This memory format-to-register format conversion includes a partial conversion process and a final conversion process. The partial conversion process includes the steps of formatting data into a format which is suitable for storage in the registers of the register file, detecting whether operand data to be operated on includes a special case exponent, and generating at least one special case flag which indicates whether or not a special case exponent has been detected.

Detailed Description Text (8):
Each of the standard MAC units 6 and 7 comprises one 82-bit adder and one 82-bit multiplier. The operands to be operated on are received by the register file block 11 from an instruction decoder (not shown) comprised by the processor architecture. The instruction decoder provides control bits to the register file block 11 along with the operands and these control bits are utilized by the MAC units to determine the type of arithmetic operation to be performed on the operands, e.g., adds, subtracts, multiplies, etc. The register file block 11 comprises a plurality of registers in which the operands received by the register file block 11 are stored.

Detailed Description Text (10):
After the operands have been written to the appropriate registers in the register file block 11, the register file block 11 reads the operands out of the appropriate registers and the register file 11 passes them to the appropriate bypass block, as indicated by the arrows on lines 20, 21 and 22 directed from the register file block 11 to the bypass block 8. The lines 20, 21 and 22 correspond to the operand bus comprised in the floating-point unit 1 and each of the lines 20, 21 and 22 corresponds to a plurality of lines needed for transporting the multi-bit operands A, B and C. The circles in FIG. 1 are intended to denote bus inputs to the blocks on which they are located. The register file block 11 reads the second set of operands A, B and C out of the appropriate registers in the register file block 11 which routes them to the appropriate bypass block, as indicated by the arrows on lines 24, 25 and 26. These lines also represent a plurality of operand bus lines.

Detailed Description Text (17):
The standard MAC units 51 and 52 preferably are very similar to the standard MAC units 6 and 7 shown in FIG. 1. In this case, the standard MAC units 51 and 52 will each comprise one 82-bit adder and one 82-bit multiplier (not shown). However, the standard MAC units 51 and 52 are each configured to receive a particular bit and to utilize this bit to cause the standard MAC units to select the appropriate half of a 64-bit word, as described below in more detail.

Detailed Description Text (20):
The lines shown in the floating-point unit 50 of FIG. 2 are being used in the same manner in which they were used in FIG. 1 to denote buses. The arrows are being used to indicate the direction of flow of the data and the circles are being used to indicate bus inputs. The lines 61, 62 and 63 represent the lower 32 bits of the SIMD word. Therefore, in SIMD mode, each of the buses 61, 62 and 63 transports a 32-bit operand (i.e., A, B and C). When the SIMD words are delivered to the floating-point unit 50, the register file block 56 loads the SIMD bits into the appropriate registers of the register file block 56 in accordance with control bits received by the register file block 56. The bypass block 54 selects the lower 32-bit portions of the SIMD words and routes the 32-bit words over buses 61, 62 and 63 from the register file block 56 to the standard MAC unit 51. Simultaneously, the bypass block 55 routes the upper 32-bit portions of the SIMD words over bus lines 65, 66 and 67 to the standard MAC unit 52.

CLAIMS:

1. A floating-point unit configured to perform load bypass operations, the floating-point unit comprising:

a register file, the register file comprising a plurality of registers for storing operand data to be operated on and for storing results of operations that have been performed by the floating-point unit;

at least one bypass component, the bypass component being configured to perform a memory format-to-register format conversion, the memory format-to-register format conversion including a partial conversion process and a final conversion process, the partial conversion process including the steps of formatting data into a format which is suitable for storage in the registers of the register file, detecting whether operand data to be operated on includes a special case exponent, and generating at least one special case flag which indicates whether or not a special case exponent has been detected;

at least one multiply accumulate unit, the multiply accumulate unit being configured to perform an arithmetic operation on the operand data, the multiply accumulate unit being configured to receive the results of the partial conversion process including said at least one special case flag and to perform the final conversion process.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC | Draw Desc | Image

☐ 4. Document ID: US 5204828 A

L9: Entry 4 of 7 File: USPT Apr 20, 1993

DOCUMENT-IDENTIFIER: US 5204828 A
TITLE: Bus apparatus having hold registers for parallel processing in a microprocessor

Brief Summary Text (10):
A bus control apparatus for performing dual arithmetic operations in a microprocessor capable of executing floating-point operations is described. The microprocessor provides first and second floating-point source instruction operands and a floating-point destination register. A multiplier means is used to multiply first and second operands to produce a first result. An adder means is used for adding third and fourth operands to produce a second result. The present invention also includes a register means for storing real and imaginary portions of a constant used for performing inner loop calculations for a given algorithm, and also for temporarily storing the first result produced by the multiplier. Data path control means are used for selecting one of a plurality of operands to be coupled to each of the operand inputs of the multiplier and the adder so as to realize a predetermined algorithm in a parallel manner. This aspect of the present invention permits a wide variety of algorithms, combining both multiply and add operations in parallel, to be implemented within the microprocessor.

Brief Summary Text (11):
Finally, interconnections are provided in the bus control apparatus for connecting the plurality of operands (which may include the first result, the second result, first and second source operands, a constant or the temporarily stored first result) to the inputs of the data path control means. For a given algorithm implementation the data path control means determine which particular operand input is to be coupled to the appropriate input of either the multiplier or the adder. By way of example, sixteen different data paths implementing sixteen different software instructions, or algorithms, are shown in accordance with the teachings of the present invention.

Detailed Description Text (9):
In operation, one operand out of a plurality of operands (shown by arrows directed into the horizontal lines representing the data-path control members) is selected to be coupled into either the multiplier or adder unit. For example, data-path control member 23, provides either the constant imaginary value stored in KI, the constant real value stored in KR or the source operand src1 to the first operand input of the multiplier unit 24 depending on which algorithm is to be implemented. In the preferred embodiment, control for each of the multiplexers 23, 25, 31 and 33 is provided by a 4-bit data-path control field (DPC) in the opcode. The DPC specifies the operands and also the loading of the special registers.

Detailed Description Text (10):
FIG. 2 shows the complete bus connection matrix used to realize all the possible algorithms supported by a preferred embodiment of the present invention. Thus, operand 1 of multiplier unit 24 is selected to be either KR, supplied from register 22, K1 from register 21 or src1 supplied along line 20. The determination of which one of these values becomes operand 1 (op1) of the multiplier is fixed by the particular encoding of the DPC. Similarly, operand 2 (op2) of the multiplier can either be src2, supplied from line 26, or the last stage result of the adder pipeline appearing on line 34. Control member 25 determines which of these two values becomes operand 2. Operand 1 of the adder can either be src1, connected from line 20, the temporary result value stored in T register 30, or the last stage result of the adder pipeline input along line 34. Control member 31 is used to select the appropriate data path for the operand 1 input of adder unit 32. Finally, operand 2 of adder 32 is selected to be either src2 from line 26, the last-stage result of the multiplier pipeline on line 27, or the last-stage result of the adder pipeline supplied on line 34. The control member, or multiplexor means, 33 is directed by the DPC to select which input operand becomes operand 2 of adder unit 32. The result provided by adder unit 32 along line 34 represents the rdest value which is coupled to one of the 32 floating-point registers of the processor.

CLAIMS:

1. A bus apparatus in a floating-point unit that includes a file of floating point registers for supplying one of a plurality of operands including a first source operand to a first source bus and a second source operand to a second source bus, said file coupled to a result bus, said floating point unit also including a pipelined multiplier having first and second operand inputs for multiplying two operands to produce a

multiplier result at a last multiplier stage and a pipelined adder having third and fourth operand inputs for adding two operands to produce an adder result that is supplied to the result bus from a last adder stage, said bus apparatus for implementing algorithms which require simultaneous add and multiply operations, said bus apparatus comprising:

a first constant holding register having a single input coupled to the first source bus, said first holding register dedicated for storing a constant;

a temporary holding register having a single input coupled to the last multiplier stage, said second holding register dedicated for storing the multiplier result;

a plurality of multiplexers including a first multiplexer means for selectively coupling into the first operand input of the pipelined multiplier a first one of a plurality of operands including the first source operand and the constant stored in said first constant holding register, a second multiplexer means for selectively coupling into the second operand input a second one of said plurality of operands including the second source operand and the result operand a third multiplexer means for selectively coupling to said third operand input in said adder means a third one of said plurality of operands including the first source operand, the multiplier result stored in said temporary holding register, and the adder result, and a fourth multiplexer means for selectively coupling to said fourth operand input in said adder means a fourth one of said plurality of operands including the multiplexer result, the adder result, and the second source operand; and

control means coupled to said first, second, third, and fourth multiplexer means for selecting which of said plurality of operands are said first one, said second one, said third one; and said fourth one of said plurality of operands.

☐ 5. Document ID: US 5001661 A

L9: Entry 5 of 7 File: USPT Mar 19, 1991

DOCUMENT-IDENTIFIER: US 5001661 A
TITLE: Data processor with combined adaptive LMS and general multiplication functions

Brief Summary Text (12):
In carrying out the above and other objects of the present invention, there is provided, in one form, a data processing system using time multiplexed circuitry to combine digital filtering and general multiplication functions with the same circuitry. The digital filtering includes implementing a predetermined algorithm in an N-tap filter, where N is an integer. A convolution of a series of data values and coefficient values for the N taps is calculated while an adaption calculation of the series of coefficient values is performed by updating the coefficient value for each tap of the filter. The general multiplication includes multiplying first and second input operands. An update portion has a first multiplier and a first adder for performing a predetermined number of update calculations to the coefficient values required to implement the predetermined algorithm. The update portion also selectively provides a first plurality of recorded operands associated with the general multiplication at an output of the first multiplier and provides partial products at an output of the first adder. A convolution portion comprises a second multiplier and a second adder coupled to the update portion. The convolution portion sums a predetermined number of convolution products, each formed by multiplying a predetermined coefficient value and a predetermined data value to provide an output signal which represents the convolution. The second multiplier also selectively provides a second plurality of recoded operands associated with the general multiplication at an output of the second multiplier and provides partial products at an output of the second adder. The second adder selectively provides an output product of the general multiplication. A storage device is coupled to both the update portion and convolution portion for selectively

providing both the data values and the coefficient values required by both the update portion and convolution portion to implement the predetermined algorithm.

Detailed Description Text (3):
Additionally, a latch 119 for receiving an operand labeled "Operand X" is connected to a data bus 170. An output of latch 119 is connected to an input of a data shifter 120 and to an input labeled "C" of multiplexor 86. Multiplexor 86 also has an input labeled "B" for receiving a zero value and an input labeled "A" connected to the latch 85. The "A" and "B" inputs of multiplexor 86 are used exclusively in the filter operation described in the incorporated copending application. A latch 121 for receiving an operand labeled "Operand Y" is connected to data bus 170. An output of latch 121 is connected to an input of a cycle select circuit 122. A first multi-bit output of cycle select circuit 122 is connected to an input labeled "A" of recorder 92. A second multi-bit output of cycle select circuit 122 is connected to an input also labeled "A" of recorder 93. A multiplexor 123 has first and second inputs respectively labeled "A" and "B" and an output connected to a first input labeled "A" of adder 96. A second input of adder 96 which is connected to an output of latch 88 is labeled "B". A multiplexor 124 has a first input labeled "A" connected to an output of shifter 120 and has a second input labeled "B" connected to the output of latch 97. A shifter 125 has an input connected to the output of latch 114 and an output connected to the first input of multiplexor 123. An output of latch 104 is connected to the second input of multiplexor 123. A latch 126 has first and second inputs respectively labeled "A" and "B" connected to first and second outputs of latch 108. A first output of latch 126 is connected to an input labeled "B" of adder 110, and a second output of latch 126 is connected to a control input of adder 110. A multiplexor 127 has a first input labeled "A" connected to the output of latch 97 and has a second input labeled "B" connected to the output of latch 114. An output of multiplexor 127 is connected to an input labeled "A" of adder 110. A control signal C4 is connected to a control input of multiplexor 86, and a control signal C5 is connected to a control input of latch 116. A control signal C6 is connected to a control input of each of multiplexors 123, 124 and 127, and a control signal C7 is connected to a control input of each of recoders 92 and 93. A control signal C8 is connected to a control input of latch 126, and a control signal C9 is connected to a control input of latch 112. A Reset signal is connected to a control input of latch 114.

Detailed Description Text (8):
Shown in FIGS. 3A and 3B is the operation of a similar multiplication operation by data processor 10 in accordance with the present invention. The operation is pipelined and may be done by the same hardware which is capable of implementing functions required by the ANSI standard to implement a U-interface transceiver. In the illustrated form, a twenty-four bit value for operand Y is shown. For this size of operand, four clock cycles each having two clock phases are required to complete a recoding operation and provide a final output product. Initially, an operand X and an operand Y are coupled by data bus 170 to data processor 10. Operand X is stored in latch 119 and shifted left by three bit positions by shifter 120 before being connected to inputs A of each of multiplexor 124 and adder 106. Operand Y is stored in latch 121 and connected to cycle select circuit 122 which couples the appropriate four bits of operand Y into recoders 92 and 93. During phase one of clock cycle one, operand X is connected to input A of multiplier 87 in response to control signal C4. Also, operand X which is shifted to the left by three bits is connected to input A of multiplier 106 in response to control signal C6. Control signal C7 allows recoders 92 and 93 to function to recode four bits of operand Y.

CLAIMS:

3. A data processing system using time multiplexed circuitry to combine digital filtering and general multiplication unctions, said digital filtering including implementing a predetermined algorithm in an N-tap filter, where N is an integer, by calculating a convolution of a series of data values and coefficient values for the N taps and substantially concurrently performing an adaption calculation of the series of coefficient values by updating the coefficient value for each tap of the filter, said general multiplication including multiplying first and second input operands, said data processing system comprising:

update means comprising a first multiplier having an output coupled to a first input of a first adder for performing a predetermined number of update calculations to the coefficient values required to implement the predetermined algorithm, said update means also selectively providing a first plurality of recoded operands associated with the general multiplication at an output of the first multiplier and providing partial

products at an output of the first adder;

convolution means comprising a second multiplier having an output coupled to a first input of a second adder, the second adder having a second input coupled to a second input of the first adder of the update means for summing a predetermined number of convolution products, each formed by multiplying a predetermined coefficient value and a predetermined data value, to provide an output signal which represents the convolution, said second multiplier also selectively providing a second plurality of recoded operands associated with the general multiplication at the output of the second multiplier and providing partial products at an output of the second adder, said second adder selectively providing an output product of the general multiplication;

storage means coupled to an input of each of the first and second multipliers of the update means and the convolution means, respectively, for selectively providing the data values and coefficient values required by both the update means and convolution means to implement the predetermined algorithm,

the first means, second means and storage means having control inputs for receiving control signals for controlling the time multiplexed circuitry performing digital filtering and general multiplication functions.

5. The data processing system of claim 3 wherein the first and second plurality of recoded operands are operands recoded either by grouping four or more bits to implement further modified Booth's algorithm, by grouping three bits to implement modified Booth's algorithm or by grouping two bits to implement Booth's algorithm.

☐ 6. Document ID: US 4996660 A

L9: Entry 6 of 7 File: USPT Feb 26, 1991

DOCUMENT-IDENTIFIER: US 4996660 A
TITLE: Selection of divisor multipliers in a floating point divide circuit

Abstract Text (1):
A multiple selector logic circuit for selecting divisor multiples in 2-bit, non-restoring divide sequences, which provides a proper and accurate quotient result and remainder, and which produces rounding and indication of exact or inexact result in conformance with ANSI/IEEE Standard 754-1985; the multiple selector logic circuit incorporates semiconductor circuits including a multiplier table having a particular matrix of multipliers which meet the standard.

Brief Summary Text (14):
The multiples are preselected by implementing a special divide algorithm which tests all possible multiples which are usable within the physical constraints imposed by the divide circuits, and which produce results which conform to the requirements of ANSI/IEEE Standard 754-1985. The preselected multiples are stored in a multiple selector logic circuit in the form of a matrix tabulation which is accessible by signal lines representing particular 3-bit combinations of dividend and divisor values.

Detailed Description Text (7):
In order to selectively control which of the plurality of inputs to the FA register 16 are to be gated into the register, the FA register 16 has an input multiplexer circuit 15 associated therewith. Multiplexer circuit 15 provides for the selective gating of the various inputs into register 16, under control of an activation signal A. Other registers in the floating point arithmetic circuit 10 have similar multiplexer input controls.

Detailed Description Text (13):
The multiple selector 24 is a logic circuit which generates multiples of the

multiplicand for the multiply operation, and multiples of the divisor for the divide operation. For the divide operation, the multiples are decoded from the high-order bits of the dividend, or partial dividend, and of the divisor. The complement of the operand may be generated by the multiple selector when necessary. The multiple selector 24 receives data from the FA register 16, the FB register 18, and the FC register 22. The multiple selector 24 has outputs to the carry-save adder 28. The multiple selector 24 contains a logic circuit which embodies a matrix of multipliers potentially usable as divisor multipliers during each iteration of a divide sequence. These multipliers are selected by access lines which are activated by decoding the upper three bits of the divisor quantity and the upper three bits of the dividend or partial dividend quantity for each iteration. Although the multipliers are embodied in physical form in the form of a semiconductor logic matrix or programmable read-only memory chip, they are represented herein in tabular form for ease of reference and understanding. The selection of appropriate multipliers is critical to the operation of the floating point divider circuit, for they must not only provide a correct quotient result, but also must provide a correct indication of the remainder which is used in the rounding operation of the divide sequence. The present invention relates to the selection of these multipliers, and particularly to the selection of multipliers which conform to the ANSI/IEEE Standard 754-1985.

Detailed Description Text (41):
The algorithms illustrated in FIGS. 2 and 3 produce a number of candidate values for M which pass the test, including value of M of zero, 1/2, 3/4, 1, 3/2, -1/2, -3/4, -1, -3/2 and -2. The foregoing candidates are chosen because each involves a relatively simple arithmetic manipulation of a binary number, such as right or left shifting, to produce fractional quantities or powers of 2, or shifting with a simple addition of another shifted value to produce a sum of two numbers, which can be relatively easily developed in standard computer hardware. Applying these candidate values of M to the multiple selector circuit 24 of FIG. 1, and more particularly to the logic circuit which is accessible during the divide iterations, leads to a logic circuit which may be represented in the form shown in Table 1. Table 1 shows a matrix of M values, wherein each value of M is found at the intersection of a particular divisor value and a particular dividend value, represented either as a true dividend or a complement dividend value. The physical logic circuit matrix is designed to produce the particular multiplier M upon simultaneous activation of signal lines representative of the particular divisor and dividend values.

Detailed Description Text (59):
Table 2 represents the electrical characteristics of a semiconductor chip which may be utilized in multiple selector 24 for deriving the correct multiple M for use in each divide iteration. In order to complete the divide operation, the quotient bits for each iteration must be selected, which is accomplished by another circuit logic chip which is preprogrammed to provide quotient bit values as an output, as a function of the values calculated during the iteration.

CLAIMS:

1. In a floating point divide circuit of the type using a 2-bit, non-restoring division method, the improvement comprising a multiplier selection circuit having means for selecting a divisor multiple as a function of the most significant bits of the divisor and dividend, said multiplier selection circuit further comprising a semiconductor circuit having a selectable matrix of signals representative of possible divisor multipliers, none of said divisor multipliers being equal to three-fourths the value of the dividend divided by the divisor for all values of dividend and divisor within the range of most significant bits of interest.

2. An apparatus for producing divisor multiples in a floating point arithmetic circuit, comprising a semiconductor circuit having a matrix of signal-producing cells therein, each of said cells being connected to sense the most significant bits of dividend registers and divisor registers, and each of said cells having means for producing a signal representative of a divisor multiplier in response to signals representative of the most significant bits of dividends and divisors, said signals representative of divisor multipliers further comprising signals representing divisor multipliers not equal to 3/4 D/V, where D and V respectively are all dividend and divisor values within the range of most significant bits to which said input selection circuit responds.

4. An apparatus for producing divisor multiples in a floating point arthmetic circuit, comprising a semiconductor circuit having a matrix of signal-producing cells therein, each of said cells connected to an input selection circuit and having means for

producing a signal representative of a divisor multiple to said input selection circuit, said matrix of signal-producing cells being responsive to signals representative of most significant bits of dividends and divisors; said matrix of signal-producing cells being constructed to produce specific divisor multiples according to the following combinations of divisors and dividends:

COMPLEMENT DIVISOR	DIVIDEND 100	101	110	111	DIVIDEND 100	101	110	111	TRUE DIVISOR
000	3/2	3/2	1	1	001	0	0	0	0
001	3/2	1	1	1	010	1/2	1/2	1/2	1/2
010	1	1	1	3/4	011	1/2	1/2	1/2	1/2
011	1	1	3/4	3/4	100	1	1	3/4	3/4
100	1/2	1/2	1/2	1/2	101	1	1	3/4	3/4
101	1/2	1/2	1/2	1/2	110	3/2	1	1	1
110	0	0	0	0	111	3/2	3/2	1	1
111	0	0	0	0					

☐ 7. Document ID: US 4862405 A

L9: Entry 7 of 7 File: USPT Aug 29, 1989

DOCUMENT-IDENTIFIER: US 4862405 A
TITLE: Apparatus and method for expediting subtraction procedures in a carry/save adder multiplication unit

Drawing Description Text (5):
FIG. 4 is a block diagram of selected elements of FIG. 1 indicating how the additional logic "1" signal required for the completion of the 2's complement of the multiplicand operand is entered in the multiplication unit according to the prior art.

Drawing Description Text (6):
FIG. 5 is a block diagram of selected elements of FIG. 1 illustrating how the additional logic "1" signal required to complete the 2's complement of the mulitplicand operand is entered in the multiplier unit according to the present invention.

CLAIMS:

6. The method of implementing a modified Booth's algorithm in a multiplier unit for multiplying a multiplicand operand by a multiplier operand, said multiplier unit having a plurality of carry/save adder stages arranged in a sequence, each of said carry/save adder stages including an array of carry/sve adder units, said method comprising the steps of:

recoding a multiplier operand to obtain control signals for each of said carry/save adder stages;

performing a sequential processing operation on said multiplicand operand in response to said control signals by said each carry/save adder stage;

combining said multiplicand operand on which said processing operation was performed with an operand from a next prior carry/save adder stage, said next prior carry/save adder stage operand being shifted relative to said multiplicand operand a preestablished number of bit positions;

when said control signals from said recoding step indicate that a subtraction operation is performed by an associated carry/save adder stage, generating a positive logic signal;

adding said positive logic signal to predetermined position of a processed multiplicand operand in a first carry/save adder stage; and

performing a 1's complement processing operation on a multiplicand operand in response to said control signals applied to a carry/save adder stage receiving control signals associated with said positive logic signal; and

combining logic signals from a last carry/save adder stage to provide a resultant operand.

Terms	Documents
L8 and l3	7

WEST

Generate Collection | Print

Search Results - Record(s) 1 through 5 of 5 returned.

☐ 1. Document ID: US 6446195 B1

L10: Entry 1 of 5 File: USPT Sep 3, 2002

DOCUMENT-IDENTIFIER: US 6446195 B1
TITLE: Dyadic operations instruction processor with configurable functional blocks

Brief Summary Text (4):
Single chip digital signal processing devices (DSP) are relatively well known. DSPs generally are distinguished from general purpose microprocessors in that DSPs typically support accelerated arithmetic operations by including a dedicated multiplier and accumulator (MAC) for performing multiplication of digital numbers. The instruction set for a typical DSP device usually includes a MAC instruction for performing multiplication of new operands and addition with a prior accumulated value stored within an accumulator register. A MAC instruction is typically the only instruction provided in prior art digital signal processors where two DSP operations, multiply followed by add, are performed by the execution of one instruction. However, when performing signal processing functions on data it is often desirable to perform other DSP operations in varying combinations.

Detailed Description Text (15):
Referring now to FIG. 5A, a block diagram of a signal processing unit 300 is illustrated which represents an instance of the SPs 300A-300D. Each of the signal processing units 300 includes a data typer and aligner 502, a first multiplier M1504A, a compressor 506, a first adder A1510A, a second adder A2510B, an accumulator register 512, a third adder A3510C, and a second multiplier M2504B. Adders 510A-510C are similar in structure and are generally referred to as adder 510. Multipliers 504A and 504B are similar in structure and generally referred to as multiplier 504. Each of the multipliers 504A and 504B have a multiplexer 514A and 514B respectively at its input stage to multiplex different inputs from different busses into the multipliers. Each of the adders 510A, 510B, 510C also have a multiplexer 520A, 520B, and 520C respectively at its input stage to multiplex different inputs from different busses into the adders. These multiplexers and other control logic allow the adders, multipliers and other components within the signal processing units 300A-300C to be flexibly interconnected by proper selection of multiplexers. In the preferred embodiment, multiplier M1504A, compressor 506, adder A1510A, adder A2510B and accumulator 512 can receive inputs directly from external data buses through the data typer and aligner 502. In the preferred embodiment, adder 510C and multiplier M2504B receive inputs from the accumulator 512 or the outputs from the execution units multiplier M1504A, compressor 506, adder A1510A, and adder A2510B.

Detailed Description Text (17):
Referring now to FIG. 5B, a more detailed block diagram of the functional blocks and the bus structure of the signal processing unit is illustrated. Dyadic DSP instructions are possible because of the structure and functionality provided in each signal processing unit. Output signals are coupled out of the signal processor 300 on the Z output bus 532 through the data typer and aligner 502. Input signals are coupled into the signal processor 300 on the X input bus 531 and Y input bus 533 through the data typer and aligner 502. Internally, the data typer and aligner 502 has a different data bus to couple to each of multiplier M1504A, compressor 506, adder A1510A, adder A2510B, and accumulator register AR 512. While the data typer and aligner 502 could have data busses coupling to the adder A3510C and the multiplier M2504B, in the preferred embodiment it does not in order to avoid extra data lines and conserve area usage of an integrated circuit. Output data is coupled from the accumulator register AR 512 into the data typer and aligner 502. Multiplier M1504A has buses to couple its output into the inputs of the compressor 506, adder A1510A, adder A2510B, and the accumulator

registers AR 512. Compressor 506 has buses to couple its output into the inputs of adder A1510A and adder A2510B. Adder A1510A has a bus to couple its output into the accumulator registers 512. Adder A2510B has buses to couple its output into the accumulator registers 512. Accumulator registers 512 has buses to couple its output into multiplier M2504B, adder A3510C, and data typer and aligner 502. Adder A3510C has buses to couple its output into the multiplier M2504B and the accumulator registers 512. Multiplier M2504B has buses to couple its output into the inputs of the adder A3510C and the accumulator registers AR 512.

Detailed Description Text (22):
The instruction set architecture of the ASSP 150 has 4 distinct types of instructions to optimize the DSP operational mix. These are (1) a 20-bit DSP instruction that uses mode bits in control registers (i.e. mode registers), (2) a 40-bit DSP instruction having control extensions that can override mode registers, (3) a 20-bit dyadic DSP instruction, and (4) a 40 bit dyadic DSP instruction. These instructions are for accelerating calculations within the core processor 200 of the type where D=[(A op1 B) op2 C] and each of "op1" and "op2" can be a multiply, add or extremum (min/max) class of operation on the three operands A, B, and C. The ISA of the ASSP 150 which accelerates these calculations allows efficient chaining of different combinations of operations. Because these type of operations require three operands, they must be available to the processor. However, because the device size places limits on the bus structure, bandwidth is limited to two vector reads and one vector write each cycle into and out of data memory 202. Thus one of the operands, such as B or C, needs to come from another source within the core processor 200. The third operand can be placed into one of the registers of the accumulator 512 or the RISC register file 413. In order to accomplish this within the core processor 200 there are two subclasses of the 20-bit DSP instructions which are (1) A and B specified by a 4-bit specifier, and C and D by a 1-bit specifier and (2) A and C specified by a 4-bit specifier, and B and D by a 1 bit specifier.

Detailed Description Text (25):
In order to access operands within the data memory 202 or registers within the accumulator 512 or register file 413, a 6-bit specifier is used in the DSP extended instructions to access operands in memory and registers. Of the six bit specifier used in the extended DSP instructions, the MSB (Bit 5) indicates whether the access is a memory access or register access. In the preferred embodiment, if Bit 5 is set to logical one, it denotes a memory access for an operand. If Bit 5 is set to a logical zero, it denotes a register access for an operand. If Bit 5 is set to 1, the contents of a specified register (rX where X: 0-7) are used to obtain the effective memory address and post-modify the pointer field by one of two possible offsets specified in one of the specified rX registers. If Bit 5 is set to 0, Bit 4 determines what register set has the contents of the desired operand. If Bit-4 is set to 0, then the remaining specified bits 3:0 control access to the registers within the register file 413 or to registers within the signal processing units 300.

Detailed Description Text (38):
The "vmul.vertline.vmuln" field refers to either positive vector multiplication or negative vector multiplication being selected as the MAIN OP. The next field, "vadd.vertline.vsub.vertline.vmax.vertline.sadd.vertline.ssub.vertline.sma x", refers to either vector add, vector subtract, vector maximum, scalar add, scalar subtraction, or scalar maximum being selected as the SUB OP. The next field, "da", refers to selecting one of the registers within the accumulator for storage of results. The field "sx" refers to selecting a register within the RISC register file 413 which points to a memory location in memory as one of the sources of operands. The field "sa" refers to selecting the contents of a register within the accumulator as one of the sources of operands. The field "sy" refers to selecting a register within the RISC register file 413 which points to a memory location in memory as another one of the sources of operands. The field of "[,(ps0).vertline.ps1)]" refers to pair selection of keyword PS0 or PS1 specifying which are the source-destination pairs of a parallel-store control register. Referring now to FIG. 6E and 6F, lists of the set of 20-bit DSP and control instructions for the ISA of the present invention is illustrated. FIG. 6G lists the set of extended control instructions for the ISA of the present invention. FIG. 6H lists the set of 40-bit DSP instructions for the ISA of the present invention. FIG. 6I lists the set of addressing instructions for the ISA of the present invention.

Detailed Description Text (39):
Referring now to FIG. 7, a block diagram illustrates the instruction decoding for configuring the blocks of the signal processing unit 300. The signal processor 300 includes the final decoders 704A through 704N, and multiplexers 720A through 720N. The

multiplexers 720A through 720N are representative of the multiplexers 514, 516, 520, and 522 in FIG. 5B. The predecoding 702 is provided by the RISC control unit 302 and the pipe control 304. An instruction is provided to the predecoding 702 such as a dyadic DSP instruction 600. The predecoding 702 provides preliminary signals to the appropriate final decoders 704A through 704N on how the multiplexers 720A through 720N are to be selected for the given instruction. Referring back to FIG. 5B, in a dyadic DSP instruction the MAIN OP generally, if not a NOP, is performed by the blocks of the multiplier M1504A, compressor 506, adder A1510A, and adder A2 S10B. The result is stored in one of the registers within the accumulator register AR 512. In the dyadic DSP instruction the SUB OP generally, if not a NOP, is performed by the blocks of the adder A3510C and the multiplier M2504B. For example, if the dyadic DSP instruction is to perform is an ADD and MULT, then the ADD operation of the MAIN OP is performed by the adder A1510A and the SUB OP is performed by the multiplier M1504A. The predecoding 720 and the final decoders 704A through 704N appropriately select the respective multiplexers 720A through 720B to select the MAIN OP to be performed by the adder A1510A and the SUB OP to be performed by the multiplier M2504B. In the exemplary case, multiplexer 520A selects inputs from the data typer and aligner 502 in order for adder A1510A to perform the ADD operation, multiplexer 522 selects the output from adder 510A for accumulation in the accumulator 512, and multiplexer 514B selects outputs from the accumulator 512 as its inputs to perform the MULT SUB OP. The MAIN OP and SUB OP can be either executed sequentially (i.e. serial execution on parallel words) or in parallel (i.e. parallel execution on parallel words). If implemented sequentially, the result of the MAIN OP may be an operand of the SUB OP. The final decoders 704A through 704N have their own control logic to properly time the sequence of multiplexer selection for each element of the signal processor 300 to match the pipeline execution of how the MAIN OP and SUB OP are executed, including sequential or parallel execution. The RISC control unit 302 and the pipe control 304 in conjunction with the final decoders 704A through 704N pipelines instruction execution by pipelining the instruction itself and by providing pipelined control signals. This allows for the data path to be reconfigured by the software instructions each cycle.

CLAIMS:

1. A signal processor for performing dyadic digital signal processing instructions having main operations and sub operations, the signal processor comprising: at least one signal processing unit including, a first multiplier and a first adder to execute a main operation of a dyadic digital signal processing instruction, a second multiplier and a second adder to execute a sub operation of the dyadic digital signal processing instruction, each of the first and second adders and the first and second multipliers having a multiplexer at its input to configure the signal processing unit to execute the main operation and the sub operation of the dyadic digital signal processing instruction, and an accumulator having registers to couple to the first multiplier or the first adder to provide operands or store intermediate results therefrom and to couple to the second multiplier or the second adder to provide an operand for the sub operation of the dyadic digital signal processing instruction and to store results of the sub operation, the accumulator having a register to couple to the buffer memory to store the digital signal processed output generated by the dyadic digital signal processing instruction.

2. The signal processor of claim 1 for performing dyadic digital signal processing instructions, the signal processor further comprising: a reduced instruction set computer (RISC) control unit and a pipeline controller to predecode the dyadic digital signal processing instruction into a plurality of preliminary instruction execution signals, and wherein the at least one signal processing unit further includes a plurality of final decoders coupled to a plurality of multiplexers, each of the first and second adders and first and second multipliers having an input multiplexer from the plurality of multiplexers to receive operands responsive to the selection by those of the plurality of final decoders coupled thereto.

8. A signal processor for performing dyadic digital signal processing instructions, the signal processor comprising: a host interface to interface to an external host computer, an external memory interface to read and write data to an external memory, clock and phase-locked loop to control the timing of operations of the application specific signal processor, a memory movement engine coupled to the buffer memory to transceive data thereto and therefrom, and at least one signal processing unit including a first multiplier and a first adder to execute a main operation of a dyadic digital signal processing instruction, a second multiplier and a second adder to execute a sub operation of the dyadic digital signal processing instruction, each of the first and second adders and the first and second multipliers having a multiplexer

at its input to configure the signal processing unit to execute the main operation and the sub operation of the dyadic digital signal processing instruction, an accumulator having registers to couple to the first multiplier or the first adder to provide operands or store intermediate results therefrom and to couple to the second multiplier or the second adder to provide an operand for the sub operation of the dyadic digital signal processing instruction and to store results of the sub operation, the accumulator having a register to couple to the buffer memory to store the digital signal processed output generated by the dyadic digital signal processing instruction, a data typer and aligner to order the bits of the operands for execution with the main operation, a third adder to add operands together, and a compressor to compress more than two operands into a pair of operands.

9. The signal processor of claim 8 for performing dyadic digital signal processing instructions, the signal processor further comprising: a reduced instruction set computer (RISC) control unit and a pipeline controller to predecode the dyadic digital signal processing instruction into a plurality of preliminary instruction execution signals, and wherein the at least one signal processing unit further includes a plurality of final decoders coupled to a plurality of multiplexers, each of the first and second adders and first and second multipliers having an input multiplexer from the plurality of multiplexers to receive operands responsive to the selection by those of the plurality of final decoders coupled thereto.

32. The signal processor of claim 31 further comprising: an accumulator having registers to couple to the first digital signal processing functional block to provide operands or store intermediate results therefrom and to couple to the second digital signal processing functional block to provide operands for the sub operation of the dyadic digital signal processing instruction and to store results of the sub operation.

☐ 2. Document ID: US 6408376 B1

L10: Entry 2 of 5 File: USPT Jun 18, 2002

DOCUMENT-IDENTIFIER: US 6408376 B1
TITLE: Method and apparatus for instruction set architecture to perform primary and shadow digital signal processing sub-instructions simultaneously

Brief Summary Text (4):
Single chip digital signal processing devices (DSP) are relatively well known. DSPs generally are distinguished from general purpose microprocessors in that DSPs typically support accelerated arithmetic operations by including a dedicated multiplier and accumulator (MAC) for performing multiplication of digital numbers. The instruction set for a typical DSP device usually includes a MAC instruction for performing multiplication of new operands and addition with a prior accumulated value stored within an accumulator register. A MAC instruction is typically the only instruction provided in prior art digital signal processors where two DSP operations, multiply followed by add, are performed by the execution of one instruction. However, when performing signal processing functions on data it is often desirable to perform other DSP operations in varying combinations.

Detailed Description Text (26):
Referring now to FIG. 5A, a block diagram of a signal processing unit 300 is illustrated which represents an instance of the SPs 300A-300D. Each of the signal processing units 300 includes a data typer and aligner 502, a first multiplier M1504A, a compressor 506, a first adder A1510A, a second adder A2510B, an accumulator register 512, a third adder A3510C, and a second multiplier M2504D. Adders 510A-510C are similar in structure and are generally referred to as adder 510. Multipliers 504A and 504B are similar in structure and generally referred to as multiplier 504. Each of the multipliers 504A and 504B have a multiplexer 514A and 514B respectively at its input

stage to multiplex different inputs from different busses into the multipliers. Each of the adders 510A, 510B, 510C also have a multiplexer 520A, 520B, and 520C respectively at its input stage to multiplex different inputs from different busses into the adders. These multiplexers and other control logic allow the adders, multipliers and other components within the signal processing units 300A-300C to be flexibly interconnected by proper selection of multiplexers.

Detailed Description Text (31):
Output signals are coupled out of the signal processor 300 on the Z output bus 532 through the data typer and aligner 502. Input signals are coupled into the signal processor 300 on the X input bus 531 and Y input bus 533 through the data typer and aligner 502. Internally, the data typer and aligner 502 has a different data bus to couple to each of multiplier M1504A, compressor 506, adder A1510A, adder A2510B, and accumulator register AR 512 of the primary stage 561. The data typer and aligner 502 also has two different data busses 551 and 552 each of which couple to each of adder A3510C and multiplier M2504B of the shadow stage 562. Typically data busses 551 and 552 deliver inputs from the delayed data registers of the data typer and aligner 502 to adder A3510C and multiplier M2504B. Also, output data is coupled from the accumulator register AR 512 into the data typer and aligner 502.

Detailed Description Text (32):
Multiplier M1504A has buses to couple its output into the inputs of the compressor 506, adder A1510A, adder A2510B, and the accumulator registers AR 512. Compressor 506 has buses to couple its output into the inputs of adder A1510A and adder A2510B. Adder A1510A has a bus to couple its output into the accumulator registers 512. Adder A2510B has buses to couple its output into the accumulator registers 512. Accumulator registers 512 has buses to couple its output into multiplier M2504B, adder A3510C, and data typer and aligner 502. Adder A3510C has buses to couple its output into the multiplier M2504B and the accumulator registers 512. Multiplier M2504B has buses to couple its output into the inputs of the adder A3510C and the accumulator registers AR 512.

Detailed Description Text (38):
These instructions are for accelerating calculations within the core processor 200 of the type where D=[(A op1 B) op2 C] and each of "op1" and "op2" can be a multiply, add or extremum (min/max) class of operation on the three operands A, B, and C. The ISA of the ASSP 150 that accelerates these calculations allows efficient chaining of different combinations of operations. Because these type of operations require three operands, they must be available to the processor. However, because the device size places limits on the bus structure, bandwidth is limited to two vector reads and one vector write each cycle into and out of data memory 202. Thus one of the operands, such as B or C, needs to come from another source within the core processor 200. The third operand can be placed into one of the registers of the accumulator 512 or the RISC register file 413. In order to accomplish this within the core processor 200 there are two subclasses of the 20-bit DSP instructions which are (1) A and B specified by a 4-bit specifier, and C and D by a 1-bit specifier and (2) A and C specified by a 4-bit specifier, and B and D by a 1 bit specifier.

Detailed Description Text (42):
In order to access operands, within the data memory 202, the registers within the accumulator 512, the register file 413 of the RISC 302, or the delayed registers of the data typer and aligner 502 of the signal processing units 300, a 6-bit specifier is used in the DSP extended instructions to access operands in memory and registers. FIG. 6C shows an exemplary 6-bit operand specifier according to one embodiment of the present invention. Of the six bit specifier used in the extended DSP instructions, the MSB (Bit 5) indicates whether the access is a memory access or register access. In this embodiment, if Bit 5 is set to logical one, it denotes a memory access for an operand. If Bit 5 is set to a logical zero, it denotes a register access for an operand.

Detailed Description Text (44):
If Bit 5 is set to 0, Bit 4 determines what register set has the contents of the desired operand. If Bit-4 is set to 1, the remaining specified bits control access to the general purpose file (r0-r15) within the register file 413. If Bit-4 is set to 0, then the remaining specified bits 3:0 control access to the general purpose register file (r0-r15) within the register file 413, the accumulator registers 512 of the signal processing units 300, or the delayed data registers of the data typer and aligners 502 of the signal processing units 300. The general purpose file (GPR) holds data or memory addresses to allow RISC or DSP operand access. RISC instructions in general access only the GPR file. DSP instructions access memory using GPR as addresses.

Detailed Description Text (45):
FIG. 6E shows an exemplary 3-bit specifier for operands for use by shadow DSP instructions only. It should be noted that in one exemplary embodiment, each accumulator register 512 of each signal processing unit 300 includes registers: A0, A1, T, and TR as referenced in FIGS. 6C and 6E. The registers A0 and A1 can be used to hold the result of multiply and arithmetic operations. The T register can be used for holding temporary data and in min-max searches like trellis decoding algorithms. The TR registers records which data value gave rise to the maximum (or minimum). When the values SX1, SX2, SY1, and SY2 are specified in the ereg fields, control logic simply selects the specified delayed data for the shadow stages of each SP without shuffling. When the values SX1s, SX2s, SY1s, SY2s are specified in the ereg fields, these values designate controls specified in a shuffle control register that determine how control logic will control shadow selectors within the data typer and aligners (DTABs) 502 of each of the signal processing units (SPs) 300 to pick delayed data held in delayed data registers for use by shadow stages of the SPs as will be discussed in greater detail later.

Detailed Description Text (86):
The "vmul.vertline.vmuln" field refers to either positive vector multiplication or negative vector multiplication being selected as the MAIN OP. The next field, "vadd.vertline.vsub.vertline.vmax.vertline.sadd.vertline.ssub.vertline.sma x", refers to either vector add, vector subtract, vector maximum, scalar add, scalar subtraction, or scalar maximum being selected as the SUB OP. The next field, "da", refers to selecting one of the registers within the accumulator for storage of results. The field "sx" refers to selecting a register within the RISC register file 413 which points to a memory location in memory as one of the sources of operands. The field "sa" refers to selecting the contents of a register within the accumulator as one of the sources of operands. The field "sy" refers to selecting a register within the RISC register file 413 which points to a memory location in memory as another one of the sources of operands. The field of "[,(ps0).vertline.ps1)]" refers to pair selection of keyword PS0 or PS1 specifying which are the source-destination pairs of a parallel-store control register.

Detailed Description Text (132):
Referring back to FIG. 5B, in the primary dyadic DSP sub-instruction of the single 40-bit extended Shadow DSP instruction, the MAIN OP and SUB OP are generally performed by the blocks of the multiplier M1504A, compressor 506, adder A1510A, and adder A2510B. The result is stored in one of the registers within the accumulator register AR 512.

Detailed Description Text (134):
For the shadow dyadic DSP sub-instruction of the Shadow DSP instruction, the MAIN OP and SUB OP are generally performed by the blocks of the adder A3510C and multiplier M2504B. The result is stored in one of the registers within the accumulator register AR 512.

CLAIMS:

5. The signal processor of claim 1 for performing digital signal processing instructions, wherein:

the primary stage includes a first multiplier and a first adder;

the shadow stage includes a second multiplier and a second adder;

each of the first and second adders and the first and second multipliers have a multiplexer at their respective inputs to configure the signal processing unit to simultaneously execute the primary and shadow digital signal processing sub-instructions, respectively; and

the at least one signal processing unit further includes an accumulator having registers to couple to the first multiplier or the first adder of the primary stage to provide operands or to store results therefrom for the primary digital signal processing sub-instruction and to couple to the second multiplier or the second adder of the shadow stage to provide operands or to store results therefrom for the shadow digital signal processing sub-instruction, respectively, the accumulator having registers to couple to a buffer memory to store digital signal processed outputs generated by the primary and shadow digital signal processing sub-instructions, respectively.

9. The signal processor of claim 5 for performing digital signal processing instructions, the signal processor further comprising:

a reduced instruction set computer (RISC) control unit and a pipeline controller to predecode the primary and shadow digital signal processing sub-instructions into a plurality of preliminary instruction execution signals; and

wherein the at least one signal processing unit further includes

a plurality of final decoders coupled to a plurality of multiplexers, each of the first and second adders and first and second multipliers having an input multiplexer from the plurality of multiplexers to receive operands responsive to the selection by those of the plurality of final decoders coupled thereto.

☐ 3. Document ID: US 6401194 B1

L10: Entry 3 of 5 File: USPT Jun 4, 2002

DOCUMENT-IDENTIFIER: US 6401194 B1
TITLE: Execution unit for processing a data stream independently and in parallel

Brief Summary Text (11):
For integer multiply, a first functional unit is a 32-bit multiplier that generates a 64-bit partial carry and a 64-bit partial sum in the first clock cycle. In the second clock cycle, a 36-bit adder contained in a first ALU adds the 32 low bits of the partial carry and sum, and a 36-bit adder in a second ALU adds the 32 high bits of the partial carry and sum. The second ALU also adds a possible incoming carry bit from the first ALU when the adders add data types with widths greater than 36 bits. The output of the two ALUs can be stored in an accumulator or in a register file as the product of two integers.

Brief Summary Text (12):
The operations for integer MAC are the same as for integer multiply, except that in the first clock cycle, a value in the accumulator which is to be added is transferred to the two ALUs. In the second clock cycle, the first and second ALUs then add the accumulator bits as well as the partial sum and carry bits to provide a result to the accumulator or register file. Therefore, both integer multiply and integer MAC are executed in two clock cycles, sharing a multiplier, two ALUs, and an accumulator.

Detailed Description Text (10):
FIG. 5 is a block diagram of data path slice 410, containing a multiplier 510, two multiplexers 520, a 36-bit ALU (FALU_36) 530, and a 72-bit accumulator 540. Multiplier 510 operates on two 36-bit operands, A and B, each containing one 32-bit, two 16-bit, two 9-bit, or four 8-bit data elements. Multiplier 510 then generates partial products, resulting in a 64-bit partial carry and a 64-bit partial sum output. One such multiplier includes a Booth decoder, a partial product generator, selection logic, and a carry-save adder employing a Wallace tree structure. The Booth decoder recodes an operand B while the partial product generator generates multiples of operand A by performing shifts and 2's complement or sign encoding operations. The selection logic selects partial products of operand A according to the recoded operand B and provides the selected partial products to the carry-save adder for addition. The adder then reduces the partial products to a partial carry and partial sum. Multiplexers 520 select whether FALU_36530 operates on the partial sums and carries or on operands A and B. Simultaneously, FALU_36530 can process data in accumulator 540, which stores data in double-length registers. According to an aspect of the present invention, FALU_36530 is broken down into smaller ALUs to perform the necessary microprocessor instructions.

☐ 4. Document ID: US 5420815 A

L10: Entry 4 of 5 File: USPT May 30, 1995

DOCUMENT-IDENTIFIER: US 5420815 A
TITLE: Digital multiplication and accumulation system

Abstract Text (1):
A multiplication system performs a series of multiplications and accumulations of plural pairs of first and second operands. The system includes first and second buses, a memory for storing the plural pairs of first and second operands, and a read buffer coupled to the memory for sequentially reading the first and second operands. An accumulator coupled to the first bus receives the first operands from the read buffer and stores the first operands. A multiplier, coupled to the first and second buses, receives the first and second operands in parallel over the first and second buses respectively from the accumulator and the read buffer respectively to provide a series of products. The system further includes an accumulator for accumulating the products to provide a final accumulated product.

Detailed Description Text (14):
The multiplier 38 comprises a 16-bit by 8-bit floating point signed magnitude multiplier of the type well known in art. It includes a first input 76 for receiving the first operands and second input 78 for receiving the second operands which have been converted to floating point format by the first converting means 36. The multiplier 38 multiplies the multiple-bit mantissas of the operands and provides a floating point product at its output 80. The multiplier further includes an adder 82 and a subtractor 84 for combining the multiple-bit exponents of the first and second operands to provide a combined exponent at an output 86.

Detailed Description Text (18):
In the first operating cycle, the first operand of the first of operands, DQ.sub.0 is read from the memory 20 and stored in the read buffer 22. In the second operating cycle, operand DQ.sub.0 is transferred from the read buffer 22 to the accumulator 30 and to the write buffer 24. This is accomplished by the read buffer 22 driving the first bus 16 to transfer DQ.sub.0 to the summer 28 through the unidirectional shift register 26. The second bus 18 is driven with all zeros so that the output of the summer 28 is the value of DQ.sub.0 which is stored in the accumulator 30. The operand DQ.sub.0 is transferred to the write buffer 24 also over the first bus 16 to prepare the write buffer 24 for transferring operand DQ.sub.0 back to the memory 20 into data location DQ.sub.1 for updating the memory 20. Also during the second operating cycle, the first operand of the second set of operands (B.sub.1) is transferred from the memory 20 into the read buffer 22.

Detailed Description Text (22):
During the sixth operating cycle, the next or second operand DQ.sub.1 of the first set of operands is transferred from the read buffer 22 to the accumulator 30 over bus 16 and through the unidirectional shift register 26 and the summer 28 in the same manner as previously described with respect to the transference of DQ.sub.0 from the read buffer 22 to the accumulator 30. Also, DQ.sub.1 is transferred from the read buffer to the write buffer. In addition, during the sixth operating cycle, the next or second operand B.sub.2 of the second set of operands is read from the memory 20 and stored in the read buffer 22.

Detailed Description Text (26):
During the tenth operating cycle, DQ.sub.2 is transferred from the read buffer 22 to the accumulator 30 in the manner as previously described and to the write buffer 24. Also, the third operand B.sub.3 of the second set of operands is read from the memory 20 and transferred to the read buffer 22.

Detailed Description Text (30):
During the fourteenth operating cycle, DQ.sub.1 is transferred from the read buffer 22 to the accumulator 30 over the first bus 16 and through the unidirectional shift

register 26 and the summer 28. Also, DQ.sub.3 is also read off of the first bus 16 and stored in the write buffer 24. To complete the fourteenth operating cycle, the last operand B.sub.4 of the second set of operands is read from the memory 20 and stored in the read buffer 22.

CLAIMS:

21. A multiplication system for performing a series of multiplications of plural pairs of first and second operands, said system comprising:

a memory for storing plural pairs of first and second operands;

read means coupled to said memory for sequentially reading said first and second operands;

first and second buses;

storing means including an accumulator coupled to said first bus for receiving said first operands from said read means and storing said first operands;

multiplier means for multiplying said pairs of first and second operands for providing a series of products, said multiplier means being coupled to said first and second buses for receiving said first and second operands in parallel over said first and second buses respectively from said storing means and said read means respectively;

summing means including a summer coupled to said first and second buses for adding said products together to provide a series of accumulated products and register means coupled to said first bus for storing each accumulated product of said series of accumulated products, said accumulator being coupled between said summer and said first bus, said accumulator temporarily storing said accumulated product and conveying said accumulated product to said register means over said first bus.

☐ 5. Document ID: US 3873820 A

L10: Entry 5 of 5 File: USPT Mar 25, 1975

DOCUMENT-IDENTIFIER: US 3873820 A
TITLE: Apparatus for checking partial products in iterative multiply operations

Brief Summary Text (10):
The above described method for detecting hardware malfunctions in the multiply unit of a digital computer utilizing an iterative addition algorithm is operable so long as the iterative addition algorithm retires only one multiplier bit per iteration. However, in response to the ever present demand for increased operating speeds, iterative addition multiply algorithms have been developed which retire a plurality of multiplier bits during each iteration. One such algorithm is disclosed in U.S. Pat. No. 3,515,344, filed Aug. 31, 1966 by R. E. Goldschmidt, et al., entitled "Apparatus for Accumulating the Sum of a Plurality of Operands," and assigned to the same assignee as this application. The multiplier disclosed by Goldschmidt, et al., is capable of retiring 6x multiplier bits per iteration, where x = 1, 2, 3, . . . , n. The multiplier bits are decoded such that x multiplier bits equal one decoded bit and the multiplicand is shifted into a series of 3-input carry-save adders (CSA) in accordance with the decoded bit. Since the CSA requires three inputs and each input is determined by x multiplier bits (one decoded bit), 6x multiplier bits are retired each iteration by the CSA's in the multiply unit.

Detailed Description Text (2):
The apparatus of the preferred embodiment of this invention is herein described in connection with a multiply unit which is substantially similar to the multiply unit disclosed by Goldschmidt, et al., U.S. Pat. No. 3,515,344, entitled "Apparatus for Accumulating the Sum of a Plurality of Operands," filed Aug. 31, 1966, assigned to the

same assignee as this invention and hereby incorporated herein by reference. FIG. 1 shows a block diagram of the residue predictor 200, actual residue generator 400 and residue comparator 300 in connection with a Goldschmidt multiply unit 100. For simplicity of description the Goldschmidt multiply unit 100 is shown in its simplest form wherein six bits of the multiplier are retired during each iteration through the adder. The above referenced Goldschmidt patent should be referred to for a detailed description of the operation of the Goldschmidt multiply unit 100 since only a working description will be presented herein.

Detailed Description Text (3):
Still referring to FIG. 1, the multiply unit 100 includes an operand input means 20, and adder tree 21, and adder loop 22, and a parallel carry propagate adder not shown. The operand input means 20 includes a multiplicand source 30, and multiplier source 31, a multiplier bit selector 31A, an iteration counter 31B, a multiplier decoder 32 and multiple gates 24. The multiple gates 24 comprise a plurality of gating devices whereby a plural binary bit operand can be gated through the devices to the input of adder tree 21. The multiplier select 31A is utilized to scan the multiplier bits and to energize multiplier decoder 32 during each iteration. The iteration counter 31B energizes the multiplier select 31A in accordance with system timing to initiate the scanning of multiplier bits stored in multiplier source 31 by the multiplier select 31A. On each iteration, 7 bits of the multiplier are examined and utilized to energize the multiplier decoder 32. On the first iteration, the multiplier select 31A is capable of transferring the first 7 bits of the multiplier to the decoder 32 from the multiplier source 31. From then on the multiplier select 31A gates succeeding groups of 7 multiplier bits to the decoder 32 in a overlapping technique such that only 6 new bits are brought into the decoder 32. On each iteration of the multiply operation, the multiplier decoder 32 will produce signals effective at multiple gates 24 to gate the multiplicand (MD) from multiplicand source 30 through the gates 24 shifted by a proper amount and/or made negative to reflect the multiple of MD dictated by the multiplier bits examined to produce at the adder tree input multiples of MD designated in FIG. 1 as M1 through M3. As in the Goldschmidt multiplier, the multiples M1 through M3 must contain some sign extension bits due to a characteristic of the CSA loop. Two sign extension bits are required in the preferred embodiment described herein. The group of signal lines labelled M1 through M3 are the multiples of the MD which are presented as operand inputs to the adder tree 21 to provide an ultimate output representing the product of the MD and multiplier bits examined. The multiplier bits are examined by the multiplier decoder 32 in overlapping groups of 3 bits in order to determine the proper amount of shift and the sign for the MD to be gated through the gates 24. In this way the high order multiplier bit from each iteration becomes the low order bit for the succeeding iteration and 6 bits of the multiplier are retired during each iteration. A summary of the multiply unit decoding technique is shown in Table 1 of FIG. 4.

CLAIMS:

4. The apparatus according to claim 3 wherein said residue prediction means further comprises a buffer means connected to said accumulator means and said comparator means for delaying the current predicted residue until the current actual residue is produced.

5. Apparatus for detecting hardware errors in an iterative addition multiply unit of a digital computer wherein the multiples for the multiply unit are determined by decoding a plurality of multiplier bits during each iteration comprising:

residue prediction means operating concurrently with said multiply unit for generating a predicted residue for partial product obtained during the current iteration, said residue prediction means including a first residue generating means connected to said multiply unit for determining the residue of the multiplicand, a second residue generating means connected to said multiply unit and to said first residue generating means for determining the residue of the current multiples to the multiply unit based on the multiplicand residue and the decoded multiplier bits, accumulator means connected to said second residue generating means for combining the current operand residues with the predicted residue for the previous iteration to yield the current predicted residue, and buffer means connected to the accumulator means for delaying said current predicted residue until the actual residue is produced;

residue generator means connected to the output of said multiply unit for determining the actual residue of the partial product generated by the multiply unit; and

comparator means connected to said residue prediction means and said residue generator

means for comparing the predicted residue with the actual residue and producing an error signal when the result of the compare is unequal.

Terms	Documents
l7 and l2	5